一种基于多重索引的大规模数据快速查找算法

应 俊[1] 杨茂斌[1,2]
(重庆邮电大学通信与信息工程学院 重庆 400065)[1] (重庆大学计算机学院 重庆 400030)[2]

摘 要 在手持式设备移动计算中,为了实时获取信息,往往需要对数据进行高效查找,而这又与手持式设备较弱的计算处理功能相矛盾。从硬件体系与软件算法综合考虑角度出发,提出了一种基于大规模记录的索引快速查找算法。实践表明,该算法结合所设计的多层次硬件体系,能高效地实现数据快速定位查找。

关键词 数据查找,索引算法,文本,大规模

Quick Search Approach to Large-scale Data Based on Multi-index

YING Jun[1] YANG Mao-bin[1,2]
(School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China)[1]
(College of Computer Science, Chongqing University, Chongqing 400044, China)[2]

Abstract To access real-time information with mobile hand processor from datum, quick search approach is adopted generally. However, there exists conflict between quick search capability demand and mobile device computing capability with weaker compute processing power. From the point of hardware architecture and software arithmetic, a quick search approach to large-scale record based on multi-index was proposed. The result shows that the proposed approach applied to the corresponding hardware architecture can increase location rate greatly.

Keywords Data search, Index arithmetic, Text, Large-scale

1 引言

随着当今IT硬件技术的不断发展,计算机CPU(Center Processing Unit)的处理能力以及存储能力早已得到空前的提高,通常情况下对非大规模数据的处理能在用户可接受的时间内得到结果。而与之伴随的数据库技术的不断发展,更是有利于大规模数据的集中管理与高效处理。然而,在移动环境中,当处理设备为以微芯片为核心所搭建的手持移动式设备,同时,又需要在较短时间内处理大量数据时,如何搭建设备的存储体系结构与设计合理的数据结构并构建高效的数据处理算法就成为核心问题。

2 数据结构与算法

2.1 数据结构

数据结构是相互之间存在一种或多种特定关系的数据元素的集合,数据元素相互之间的关系称为结构。根据数据元素之间关系有不同特性,通常有下列4类基本结构:(1)集合:结构中的数据元素之间除了"同属于一个集合"的关系外,别无其它关系;(2)线性结构:结构中的数据元素之间存在一对一的关系;(3)树形结构:结构中的数据元素之间存在一对多的关系;(4)图状结构或网状结构:结构中的数据元素之间存在多对多的关系。

数据结构不仅仅是一组数据项的组织或者结构,它同时还定义并体现了一类普通数据的表示及其相关操作。

2.2 算法

算法是为了解决某个问题而设计的一种方法,是对特定问题求解步骤的一种描述。一个算法通常具有以下5个重要特性:确性、确定性、有限性、可终止性及具体步骤等。一个问题可以有多种算法,一个给定的算法可以解决一个特定的问题。

而算法的设计要求中一个重要的方面就是要考虑高效率与低存储量需要。对同一个问题如果有多个算法可以解决,执行时间短的算法效率就高。存储量需求是指算法执行过程中所需要的最大存储空间。效率与低存储需求这两者都与问题的输入规模有关。评价这二者的度量一般记作:$T(n)=O(f(n))$,$S(n)=O(f(n))$,通常称之为算法时间复杂度与算法空间复杂度,其中,n 为输入规模大小,通常指记录条数或数据量大小。

由上可见,一个问题可以有多种数据,同时也可以有多种算法。不同的数据结构其可进行的操作方式可能不一样;不同的算法其执行的时间效率与空间存储需求也可以不一样。因此,针对某一问题我们在求解决方案时不仅仅要考虑数据结构的设计问题,同时也要考虑算法问题,以期能获得一个在时间复杂度与空间复杂度两方面的一个综合高效指标。

算法的时间高效性与空间低存储要求通常为一对关存的矛盾对立统一体,要么以时间换空间,要么以空间换时间,第一种情况即为当处理大规模数据而空间又不足时,其处理时间将变长,第二种情况为给出足够的数据处理空间,则处理时间将缩短。

3 硬件体系结构

作为一个完整的、可独立处理数据并储存数据的设备,至少要包含数据处理单元DPU(Data Process Unit)(一般为运算器或运算器与控制器所构成的整体,后者通常称为CPU)

以及存储单元 SU(Storage Unit)两大部分。通常我们希望运算器速度足够快、存储容量足够大,同时存取速度也足够快、价格也便宜,但这些指标之间是相互制约的,特别是后三者往往存在矛盾。为了能较好地解决这些矛盾,结合有效算法,往往采用图 1 所示的多级存储体系结构,最终结果就是取得一个综合性价比指标。

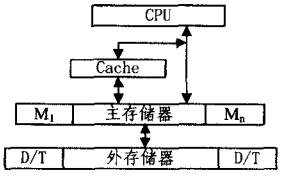

图 1　三级存储体系结构示意图

如图 1 所示,CPU 可直接按地址访问主存与 Cache,Cache 与主存之间的数据存在着映象,而主存内的数据也与外存储器的数据存在着映象。

一般而言,CPU 与主存储器(Memory)二者之间的速度存在大约一个数量级的差距,为了解决 CPU 与主存之间的速度匹配,往往在这二者之间设置一种高速缓冲存储器(Cache),其内存放着最近要使用的程序与数据,作为主存中当前活跃数据的副本。当 CPU 访问主存时,同时访问 Cache,通过对地址码的分析来判断从 Cache 读取数据还是从主存里读取数据,同时,采用某种算法不断刷新 Cache 里的数据,从而可使 CPU 更为快速地访问主存中的数据,进而访问到外存储器中的数据。

因此,当考虑一个待求解问题时,需要从软、硬件方面进行综合考虑,根据实际应用环境,设计最优化方案。

4　方案实现

基于以上理论分析,在具体的实现过程中,从系统级分析出发,进行了硬件架构设计方面的考虑,以及数据结构方面设计的考虑,具体如下。

4.1　硬件设计

针对具体的项目,而且又是可移动设备而言,本次实施采用的 ARM9 芯片,是一款较低级别的计算芯片,而所需要处理的数据却最高达到千万条记录级别,因此,硬件体系架构的设计就成为必须考虑的范围。

目前体系架构中的计算存储结构如图 2 所示。

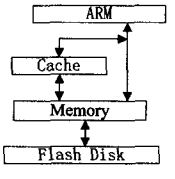

图 2　实施方案存储体系结构示意图

在该三级体系结构方案中,将 Flash Disk 作为外存储器,其目的是为了舍去耗时较多的磁盘 I/O 操作,也更适合于移动环境。在设计相应高效算法的设计基础上,充分利用各层次的性能优势,最终达到优化综合性能指标。

4.2　算法设计

一般而言,文件的体积都大于主存储器的容量,那么 CPU 对文件的读写访问就涉及到对磁盘的 I/O 操作。通常情况下,磁盘的 I/O 操作是所有操作中最为耗时的操作,这就要求组织合理的文件内容结构。

(1)文件的组织

随机访问(Random Access)算法在处理文件记录内容时独立于文件中记录的逻辑顺序,顺序访问(Sequential Access)算法按文件中记录的逻辑顺序处理记录。

如果把文件的磁盘物理布局处理为文件记录所要访问的顺序逻辑布局,那么顺序访问需要的时间将会大大减少。

在具体实现中,将文件记录的长度设计为固定大小,每条记录有一个唯一的码,称之为记录的关键码,通过它可以唯一地确定一条记录。按码的大小将记录按递增或递减进行逻辑顺序排列,并存储在磁盘连续地址上,这样可建立顺序排序的逻辑文件,并建立磁盘物理地址与记录号逻辑顺序的对应关系,如图 3 所示。

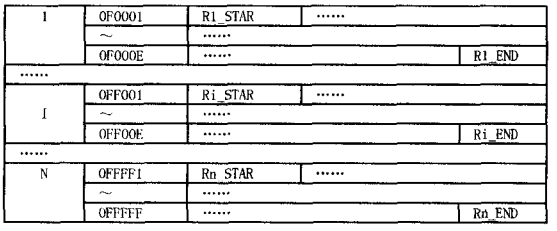

图 3　记录顺序存储结构示意图

图 3 中的 Ri_STAR 头部中包含对应该条记录的关键码。Ri_END 则为该记录的尾部,包含对应该条记录的结束标示(尽管有无该结束标志已变得不重要,但可保留以备它用)。

(2)索引建立

索引即为通常意义上的目录,在计算机数据处理中通常用来指示物理记录和逻辑记录之间的一种对应关系,其表现形式一般为索引表或索引文件。索引文件包括文件数据区和索引表两大类,索引表中的每一项称作索引项,它能常按关键字(或逻辑记录号)顺序排列。

首先,建立数据文件,如(1)中所示,表 1 为索引表示。

表 1　索引文件数据区示意

物理记录号	关键码	记录其它内容
101	01	……
103	02	……
105	03	……
107	04	……

在上述文件数据区中,如果其中的记录按关键码顺序排序,则称索引顺序文件,否则,称其为索引非顺序文件。

其次,对被索引的数据建立建立索引表。

如表 2 所列,如需第一条记,可以通过输入关键码"01"找到相应的物理记录号,再根据"101"进行直接寻址可找到该条记录的起始地址,经过对起始地址进行固定长度的偏移,可获取整条记录的内容。

表 2　索引表示意

物理记录号	关键码
101	01
103	02
105	03
107	04

索引表通常在索引数据文件的过程中建立，其建立后的初始一般为无序状态，表中的索引项按记录输入的先后次序排列，待全部记录输入完毕后再对索引进行排序。

索引文件的检索方式为直接存取或按关键字(进行简单查询)存取，检索过程一般分两步执行：首先，查找索引表，若索引表上存在该记录，则根据索引项的指示读取外存上该记录；否则说明外存上不存在该记录，也就不需要访问外存，并返回提示。

由于索引项的长度比文件中的记录小得多，一般情况下可以将索引表一次读入内存，由此在索引文件中进行检索时只访问外存两次，即一次读索引，一次读记录。并且由于索引表是有序的，则查找索引表时可采用高效的折半查找算法(Binary Search)(也有称之为二分查找算法)。

(3)二重索引

当处理文件为大规模记录时，按(2)中方法所建立的索引文件可能仍会超过主存储器的容量，也就是说主存储器不能一次性完全载入整个索引文件，这时索引文件的建立并不能有效解决记录的定位问题。

针对这种情况，根据索引思想，可以建立基于索引的索引，如表 3、表 4 所列。

表 3　一重索引

物理记录号	二重索引号
101	
103	B01
105	
107	
……	B02

表 4　二重索引

二重索引号	关键码
B01	01～03
B02	04～06
B03	07～09
B04	10～12

为了高效利用折半查找法，二重索引也需如(2)中所介绍的方法一样最终建立为顺序文件。当输入为关键码时，可利用折半查找法快速找到对应的二重索引号，并在此基础上限定物理记录号，也就是外存的地址范围，在锁定的外存在址范围里查找与关键字相等的记录关键字，从而可以快速定位一条确定的记录。

其过程为：设输入的关键字为“02”，利用二分查找法，找到其对应的二重索引号“B01”，在一重索引表上找到对应的物理记录号为 101～105，设记录长度为 n，则可用顺序查找算法或折半查找算法在 101～105 中查找，以顺序查找为例：第一次，定位 101 地址，读取记录关键字，与输入关键字进行比较，如相等则返回该记录；否则进行地址偏移，其偏移量为记录固定长度 n，重复以上步骤，直到找到为此。采用折半查找算法雷同。

(4)多重索引

根据二重索引思想，当被查找的文本所包含的记录量非常大，而所应用的相应硬件环境确定后，可采用通用算法，自动建立多重索引，直到最高重索引可以一次性完全载入主存储器中为止。图 4 是多重索引示意图。

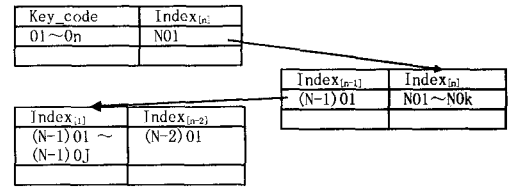

图 4　多重索引示意图

表 5 列出多重索引第一级。

表 5　多重索引第一级

Index[1]	物理记录号
	101
I01	102
	……

通过以上索引的建立，可以获得对最终数据文件记录的快速定位，并在较短时间内获得该记录。这种思想也充分利用了硬件体系的多层次体系结构的优点。

(5)算法描述

Step 1　建立索引文件

①Write Sequential Text //将文件组织为顺序文件

②Initial Memory Space //根据文件大小清空磁盘连续地址空间(需考虑索引文件体积大小)

③Load Text //顺序载入文件

④Build Index //建立一级索引

⑤If Size(Index)＞ Size(Memory/Cache)

Then ④ //比较所建索引文件的尺寸大小与主存储器或 Cache 容量大小，并依此类推建立多重索引

Step 2　执行查找

①Input Key Code (KeyS)

②Binary Search Index_n: Find(Key Code)

//n 为索引重数

③If Find(key code)= Flase Return

ElseIf n <> 1 Then n=n－1, Return ②

Else ④

④Read Physical Adress

⑤Read Record

4.3　算法分析

在实施方案中，上一节(5)算法描述中的 Step 1 可利用外部计算机处理形成，故可以不考虑其时间复杂度的问题。

如果要使用所设计的硬件设备本身来完成，则相应的算法会根据实际情况作出一些调整，在本文中主要论述对文件的索引。

以下所讨论的时间复杂度分析与空间复杂度分析都是基于 Step 2 所进行的。

(1)时间复杂度分析

Step 2 主要为折半查找与比较，折半查找时间复杂度为：$\lfloor \log_2^n \rfloor + 1$，平均查找长度为 $ASL_{bs} = \log_2^{(n+1)} - 1$，比较时间复杂度为 O(1)，其中 n 为输入规模(或记录的条数、关键字的个数)。

设一共有 m 重索引，则共需时间为 $\lfloor \log_2^{|m|} \rfloor + \lfloor \log_2^{|m-1|} \rfloor + \cdots + \lfloor \log_2^{|1|} \rfloor + m$，其中：$|m|$ 表示第 m 级索引的规模。其

(下转第 290 页)

具有较高的色彩保真度，可显著地增强图像暗区内细节信息的可视度，大大地提高简牍考古工作者史料解读工作的准确性和可靠性。

参 考 文 献

[1] Land E. An alternative technique for the computation of the designator in the retinex theory of color vision. Proc. Nat. Acad. Sci. ,1986,83:3078-3080

[2] Land E. Recent advances in retinex theory and some implications for cortical computations. Proc. Nat. Acad. Sci. ,1983,80:5163-5169

[3] Rahman Z , Jobson D J , Woodell G A . Retinex Processing for Automatic Image Enhancement // Human Vision and Electronic Imaging VII, SPIE Symposium on Electronic Imaging. Porc. SPIE 4662. 2002

[4] Kimmel R , Shaked D , Elad M . A Variational Framework for Retinex

[5] Devir Z, Tridensky L. Dynamical Range Compression While Improving the Visual Quality of Color Images

[6] Land E . Recent advances in retinex theory . Vision Research , 1986,26(1):7-21

[7] 孙炯，邵枫，陈僖雄. 一种新颖的多视点图像规正算法. 计算机工程与应用，2007(17)

[8] 刘瑛，王绪本. 简牍图像文字切分算法研究[J]. 科学技术与工程，2007,7(21):5586-5589

[9] 覃庆炎，王绪本，蒋维. 反锐化掩模法在简牍文字增强中的应用[J]. 微计算机信息，2008,3(24):241-242

[10] 芮义斌，孙锦涛，程风雷. MSR 在红外图像增强中的应用. 光电技术应用，2006(6)

[11] 王彦臣，李树杰，黄廉卿. 基于多尺度 Retinex 的数字 X 光图像增强方法研究. 光学精密工程，2006(2)

[12] 陈雾. 基于 Retinex 理论的图像增强算法研究. 学位论文. 南京理工大学，2006

(上接第 260 页)

I/O 操作实施于 $m-1$ 个索引文件及数据文件上，其 I/O 操作次数为：

$$T_I=\lfloor \log_2^{|m-1|} \rfloor+\cdots+\lfloor \log_2^{|1|} \rfloor+m-1$$

如果直接对磁盘文件进行顺序查找，则时间复杂度为 $T_S=O(n)$，也就是要进行 n 次 I/O 操作。

在设计时需精确计算 T_I 与 T_S 的关系，并依此确定 m 的大小，从而反过来给硬件设计确定相应的主存大小，通常 m 取值为 2～3。

(2)空间复杂度分析

假设数据文件记录数为 n，每记录大小为 k 字节，每记录关键字为 1 字节(仅为表述方便)，则该文件有 n 个关键字，文件总大小为 nk 字节。

假设每级索引文件的索引号所占字节数为 1，递减级数为 i，即每个索引号对应 i 个记录或次级索引号，则第一级索引文件的索引号为 n/i 条，共 $2n/i$ 个字节，第二级索引文件共 n/i^2 条记录，共 $2n/i^2$ 个字节，第三级索引文件共 n/i^3 条记录，共 $2n/i^3$ 个字节，其余依此类推。因此，索引文件的总体积相对于原数据文件以指数级下降。建立索引后的总体积大小为：

$$nk+2n(1/i+1/i^2+1/i^3+\cdots)=nk+\frac{2n(1-\frac{1}{i^n})}{i-1}$$

若 i 越大，则其索引文件的体积也就越小。

5 实现结果

在具体实现中，采用的 CPU 芯片为 ARM 芯片，设备尺寸总大小为 4cm×12cm，为可移动设备。

数据文件中每记录长为 125 字节，当输入规模为 100 万条记录时，按传统的无索引顺序查找算法，所耗时间为 3.25 秒。采用二重索引结构后，其查找时间为 0.412 秒，比率为 10。当输入规模为 1000 万条记录时，无索引顺序查找耗时为 31 秒，采用二重索引结构后，其查找时间为 1.3 秒，比率为 24。表 6 给出二重索引实验数据。

表 6 二重索引实验数据

规模	顺序查找	二重索引	比率
100 万	3.25	0.412	10
……	……	……	
1000 万	31	1.3	24

从以上统计数据分析结果表明，在一定的使用环境下，硬件体系结构与软件体系结构影响着整个数据处理的效率。

结束语 本次方案与具体实现算法已成功应用于国内某特大型国营企业相应项目的实现，目前，已良好运行近一年的时间。在将来会面临更大规模的数据处理与更高要求的移动处理，需要更进一步地优化硬件体系，同时，也需要更进一步地优化软件体系架构与算法设计。

参 考 文 献

[1] Guttman A . R - trees : a dynamic index structure for spatial searching[J]. ACM SIGMOD Record Archive, 1984,14(2):47-57

[2] Robinson J T . The K - D - B - tree : a search structure for large multidimensional dynamic indexes // Proceedings of the 1981 ACM SIGMOD International Conference on Management of Data. Ann Arbor, Michigan April 29-May 01, 1981

[3] Song S I, Kim Y H, Yoo J S. An Enhanced Concurrency Control Scheme for Multidimensional Index Structures. IEEE Transactions on Knowledge and Data Engineering,2004,16(1): 97-111

[4] Moses S, Gruenwald L. Scalability of indexing structures in a production systems testbed for computational research. Computers and Industrial Engineering, 2005 ,48(1):97-109

[5] Salton G , Buckley C . Parallel text search methods . Commun . ACM,1988,31(2):202-215

[6] Stanfill C,Thau R,Waltz D. A parallel indexed algorithm for information retrieval. Technical Report DR90-2. Thinking Machines Corporation, Cambridge, Mass. , 1990

[7] Stanfill C, Thau R, Waltz D. A parallel indexed algorithm for information retrieval. ACM SIGIR Forum, June 1989,23(SI): 88-97

[8] Jaggar D. Arm Architecture And Systems. Micro, IEEE Publication Date,1997,17:9-11

[9] Seal D. ARM Architecture Reference Manual, 2nd edition. Addison-Wesley Longman Publishing Co. , Inc. Boston, MA, USA ,2000

[10] 周立功，等. ARM 微控制器基础与实战 (第 2 版). 北京航空航天大学出版社，2005